HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

2857 AF IFW

PATENT APPLICATION

ATTORNEY DOCKET NO. 10010357-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): John S. Eden

Confirmation No.:

Application No.: 09/880,329

Examiner: Jeffrey R. West

Filing Date: 06/12/2001

Group Art Unit: 2857

Title: Flexible, Extensible, and Portable Testing Platform

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in ***triplicate*** is the Appeal Brief in this application with respect to the Notice of Appeal filed on ***April 12, 2004***.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $330.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( )	one month	$110.00
( )	two months	$420.00
( )	three months	$950.00
( )	four months	$1480.00

( ) The extension fee has already been filled in this application.

(***X***) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account ***08-2025*** the sum of ***$330.00***. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(***X***) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: ***June 14, 2004***

OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number_____ on _____

Number of pages:

Typed Name: ***Joanne Bourguignon***

Signature: ***Joanne Bourguignon***

Respectfully submitted,

John S. Eden

By ***Robert W. Bergstrom***

Robert W. Bergstrom

Attorney/Agent for Applicant(s)
Reg. No. ***39,906***

Date: ***June 14, 2004***

Telephone No.: ***206.621.1933***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Applicant: John S. Eden
Application No.: 09880,329
Filed: June 12, 2001
Title: Flexible, Extensible, and Portable Testing Platform

Examiner: Jeffrey R. West
Art Unit: 2857
Docket No.: 10010357-1
Date: June 14, 2004

BRIEF ON APPEAL

Commissioner of Patents and Trademarks
Washington, DC 20231

Sir:

This appeal is from the decision of the Examiner, in an Office Action mailed on February 12, 2004, finally rejecting claims 1-22.

REAL PARTY IN INTEREST

Hewlett-Packard Development Company, L.P. is the Assignee of the present patent application. Hewlett-Packard Development Company, L.P., is a Texas corporation with headquarters in Houston, Texas.

RELATED APPEALS AND INTERFERENCES

Applicants' representative has not identified, and does not know of, any other appeals of interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-22 are pending in the application. Claims 1-22 were finally rejected in the Office Action dated February 12, 2004. Applicant appeals the final rejection of claims 1-22, which are copied in the attached Appendix.

STATUS OF AMENDMENTS

No Amendment After Final is enclosed with this brief. The last amendments to the claims were made in the Amendment filed June 12, 2003.

SUMMARY OF INVENTION

Applicant's claimed testing platform includes a core test execution engine 212 and components that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of a computing resource environment, including an operating system interface, tested by the test engine. The test execution engine 212 interfaces to (1) data output and data storage devices 204-205 via a result handler component 214; (2) operating-system-provided functionality 202 via various components diagrammed together in Figure 2 as a generalized operating system adaptor 216; (3) user I/O devices 206-207 via a user I/O component 218; (4) a test routine via a mode component 220, a sequencer component 222, and a test executor component 224; (5) a result handler component via a first test link component 226; and (6) an operating system adaptor, user I/O handler, and a communications interface 228 via a second test link component 230. The communications interface component 228 serves to interface the test routine 210 with a hardware component 208 tested by the test routine. Figure 2 clearly shows that Applicant's test platform is a self-contained, logically isolated group of software components external to, and apart from, the operating system of the computing resource environment in which the testing platform runs. The highly modular and onion-like layers of shielding provided by the functional components of the testing platform allow for high levels of modifiability, adaptability, and portability of both the testing platform and of test routines developed to test various hardware and software components. The testing platform, for example, can be ported to almost any computing resource environment, including to many different operating systems and computer platforms, without the need to modify either the internal execution loop within the test execution engine or functional components such as the test sequencer.

ISSUES

1. Whether claims 1-22 are unpatentable under 35 U.S.C. § 103 over Liu et al., U.S. Patent No. 6,009,41 in view of Shankman, U.S. Patent No. 6,381,656 B1.

GROUPING OF CLAIMS

Claims 1-22 stand or fall together with respect to Issue 1. The default grouping of claims naturally by the claims included in each separate rejection is accepted.

ARGUMENT

Claims 1-22 are currently pending in the Application. In an office action dated February 12 25, 2004 ("Office Action"), the Examiner rejected claims 1-22 under 35 U.S.C. § 103(a) as being unpatentable over Liu et al., U.S. Patent No. 6,009,541 ("Liu") in view of Shankman et al., U.S. Patent No. 6,381,656 ("Shankman"). Applicant's representative respectfully traverses these 35 U.S.C. § 103(a) rejections.

ISSUE 1

Whether claims 1-22 are unpatentable under 35 U.S.C. § 103 over Liu et al., U.S. Patent No. 6,009,41 in view of Shankman, U.S. Patent No. 6,381,656 B1.

Consider claim 1:

> 1. A testing platform within a computing resource environment, the testing platform comprising:
> a test execution engine that receives input commands and initiates processing of the input commands;
> a test routine; and
> components that serve to adapt hardware and software interfaces of the computing resource environment to the test execution engine and test routine and that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface.

The components specified in the third element of claim 1 are clearly claimed to "shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface."

Liu

Liu discloses:

> ... a diagnostic test in conjunction with a BIOS test routine. A BIOS testing routine is initiated. Control is transferred to a diagnostic routine. The diagnostic routine performs a series of tests in which a plurality of components are examined. For example, in one embodiment the diagnostic routine tests a central processing unit, a peripheral component, an interconnect system, memory and system power. Subsequently, control is transferred to the BIOS testing routine.
>
> In one embodiment, the diagnostic routine tests memory by creating a temporary table of data from stored pseudo random data. Memory is logically divided by determining a set of address blocks which specify a set of continuous address arrays. For ease of implementation, a total number of address blocks in the set of address blocks may be equal

> to a total number of entries on the temporary table. Data is written from the temporary table into each address block of the set of address blocks in a pseudo random order according to data from the temporary table. Data stored in the memory under test is compared to data written in the memory under the test.
>
> In yet another embodiment the diagnostic test may detect a user interrupt during performance of the diagnostic routine. In response to the user interrupt, a manual diagnostic mode is initiated in which user input may be used to perform further diagnostic tests. (Liu, column 2, lines 19-45)

Liu's diagnostic test is in no way shielded from hardware and operating-system interfaces. It detects interrupts, addresses memory directly, and directly tests a CPU, system power, an interconnect system, and other components. Liu's extended BIOS testing diagnostic routine "is a separate block of code stored within a common ROM with the BIOS test routine. In another embodiment, the extended diagnostic routine is stored in a hard disk drive" (column 3, lines 9-13). In one embodiment, "the diagnostic routine begins by reading to and writing from the central processing unit (CPU). The routine may check several standard CPU registers which record errors ..." (column 3, line 67 – column 4, line 3). In one embodiment, "the diagnostic routine also performs a test of the Peripheral Component Interconnect (PCI) cards and buses or other type of bus components (column 4, lines 6-8). Therefore, as also noted by the Examiner, Liu most explicitly and decidedly does not include "components that serve to adapt hardware and software interfaces of the computing resource environment to the test execution engine and test routine and that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface," as clearly claimed in claim 1. Liu teaches exactly an opposite strategy, with Liu's diagnostic routine – a simple, unshielded code block – directly accessing hardware components such as CPU registers, PCI cards, and buses. In Applicant's representative's viewpoint, Liu is completely unrelated to the claimed invention. Indeed, Applicant's testing platform can be furnished with a test routine that test particular hardware components, but Applicant's test routine interfaces to layers of tiered, hierarchical shielding components, rather than directly to CPU registers, buses, and other computer resources within the computer-resource environment in which Applicant's testing platform happens to be running.

Liu's diagnostic routine runs in conjunction with a BIOS test routine. A BIOS is defined, in part, in the Microsoft Computer Dictionary, as an "acronym for basic input/output system, a *set of routines that work closely with the hardware* to support the transfer of information between elements of the system, such as memory, disks, and the

monitor" (emphasis added). A BIOS runs directly on hardware, below an operating system. During a system boot, the BIOS runs first, prior to any operating system. Indeed, in Figure 1 of Liu, and in the text explaining Figure 1, beginning on line 46 of column 3, Liu clearly and correctly shows that the diagnostic routine runs directly after a cold boot, and only after a cold boot, long before an operating system is loaded or executed. The term "BIOS" is well understood in almost every art relating to computers and computing.

In the last paragraph of page 3 extending to page 4 of the Office Action, the Examiner states that, "[a]lthough not specifically disclosed, it is considered inherent that the invention of Liu executes the aforementioned test routine using a corresponding operating system because the invention of Liu does disclose implementing the invention within a general purpose or specific function computer and general purpose and specific function computers require an operating system for carrying out its functions." First, Liu explicitly shows that the diagnostic routine runs directly after a cold boot, long before an operating system is loaded and executed. It is quite well known in computing that BIOS diagnostic routines must run prior to execution of the operating system, because they would otherwise interfere with and almost certainly crash the operating system. A diagnostic routine cannot write values in memory, field interrupts, and do all of the things that Liu explicitly describes his test routine as doing, in an environment in which an operating system is executing. General purpose computers, including PCs, can run a BIOS and BIOS-associated diagnostics quite well without an operating system. Any PC user can interrupt the hard boot process and interact directly with the BIOS, with no operating system yet loaded and launched.

In Applicant's representative's respectfully offered opinion, Liu is completely unrelated art. Liu's diagnostic routine is not shielded from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface. It need not be shielded from the operating system, since no operating system has been launched at the time Liu's BIOS-associated test diagnostic runs. It is not also shielded from hardware interfaces. The BIOS is more intimately associated with the hardware of a computer than any other software that runs on the computer. That is its purpose. Applicant's representative's respectfully offered opinion, the Examiner has not understood the teaching of Liu, nor the term "BIOS."

Shankman

Shankman provides a method and system for monitoring performance of I/O processors included within server hardware by introducing a virtual monitor diagrammed in Figure 4 into each I/O processor, and including an I/O monitor, diagrammed in Figure 5, within the server to collect I/O messages from the virtual monitors, and reporting performance data to an I/O user interface (Abstract of Shankman, Figure 4, and Figure 5). As stated in the Summary, Shankman's virtual-monitor-and-I/O-monitor system "monitors message traffic passing through the input/output ("I/O") processors within servers in a computing system in order to analyze the I/O processing within the computing system as a whole" (column 3, lines 1-5). Shankman's virtual-monitor-and-I/O-monitor system is also not shielded from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface. Consider, for example, Figure 5 of Shankman and the accompanying text beginning at line 3 of column 8:

> FIG. 5 is a block diagram depicting the I/O monitor 201 shown in FIG. 2 and its interfaces to other elements of the monitoring system, according to an embodiment of the invention. The I/O monitor interfaces with the I/O user interface 220 running on the workstation 200, the virtual adapters 204-206 and with server devices 501 on the server 101. The I/O monitor 201 interfaces with the server devices 501, including both I_2O devices 502 and non-I_2O devices 503.

As shown in Figure 5, and explicitly stated in the above text, Shankman's I/O-monitor directly interacts with hardware devices. Moreover, Shankman's virtual monitor directly interacts with the operating system of the I/O processor in which it runs, as clearly shown by the double arrow interconnecting the communications module of the virtual monitor 402 and the I/O processor operating system 430 in Figure 4. There is no shielding of Shankman's virtual-monitor-and-I/O-monitor system from hardware dependencies – instead, components of Shankman's virtual-monitor-and-I/O-monitor system are directly embedded into the computer system components and directly interface with them.

To refute Applicant's representative's position, beginning in the last paragraph of page 8 of the Office Action, the Examiner references column 2, lines 11-16 that describe operation of the I_2O I/O subsystem architecture. This is a hardware architecture, and is not part of Shankman's virtual-monitor-and-I/O-monitor, as clearly detailed in the paragraph of Shankman beginning on line 62 of column 1. Shielding the CPU from I/O subsystem interrupts has nothing at all to do with shielding of a software routine from hardware and operating system dependencies. It is simply a hardware architecture model. The Examiner

then states that the cited passages, from the Background of the Invention section of Shankman, indicate "that the 'IOPs' are components that adapt hardware and software interfaces of the resource environment to the monitoring device to shield the monitoring device from dependencies on the hardware and software interfaces of the computing resource environment." IOPs are defined on 20 of column 4 to be I/O processors. They are hardware devices within servers. As explicitly stated by Shankman on lines 59 and 60 of column 4, Shankman's virtual adapters monitor I_2O messages processed by IOPs. Shankman's virtual adapters are inserted into, and directly monitor hardware devices. The virtual monitors then communicate the I_2O message data to the virtual monitor, as stated beginning on line 11 of column 6. Moreover, Shankman's I/O monitor discovers and gathers statistical data on non-I_2O local area network ("LAN") devices and block storage devices on the server, as stated beginning on line 28 of column 8. Shankman's I/O monitor is thus not at all shielded from hardware interfaces, but interfaces directly to hardware. The Examiner further states that "the virtual monitor 402 is connected to the operating system 430, but the operating system 430 is an operating system of the 'IOP'/adapter." Claim 1 is directed to shielding of a test engine from operating system and hardware dependencies, not to shielding from operating systems other than IOP operating systems or hardware dependencies other than the bulk of the hardware in a computer system. In Applicant's representative's respectfully offered opinion, the Examiner has confused the term "IOP" in Shankman with Shankman's virtual adapters, which are inserted into IOPs. In fact, Shankman specifically states, on lines 34-36, that his "virtual adapter's DDM programming code may be linked directly into the resident IxWorks operating system on an IOP board on which it will function." In one statement, Shankman makes it clear that the IOP is a hardware device, and that the virtual monitor interfaces directly with the operating system of that device. Shankman further states that:

> "[t]he I/O monitor 920 preferably comprises an IBM-compatible personal computer having a Pentium processor with a minimum sped of 166 MHz, a peripheral component interconnect ("PCI") local bus, and a network interface adapter. The I/O user interface 930 preferably utilizes an IBM-compatible personal computer having a Pentium-II processor with a 266 MHz minimum speed, a network adapter, and a WINDOWS NT operating system. The I/O user interface 930 includes a WINDOWS NT graphical user interface ("GUI") application program that accesses the virtual adapter's IxWorks DDM through standard WINDOWS NT device functions which can read the stored I2O message data from the virtual adapters. (column 20, lines 20-33)

Thus, each and every component of Shankman's I/O monitoring system interfaces directly and specifically with both an operating system and with hardware.

CONCLUSION

Neither Liu nor Shankman teach, suggest, mention, or allude to test execution engines that are shielded "from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface." Liu's BIOS-associated test routine runs on a bare computer, following a hard boot, prior to operating system launch. It is associated intimately with the hardware interface of the computer on which it runs. It cannot be portable, because it directly interfaces with memory, the CPU, and other hardware components. BIOS software is, in fact, hardware specific software that allows an operating system to assume certain I/O primitives. Liu's testing routine is shielded neither from operating system nor from hardware dependencies. Shankman's virtual adapters and I/O monitor are intimately interfaced with both hardware and with the operating system. Applicant's representative would normally cite pertinent statutes and case law concerning the elements of a *prima facie* case of obviousness, but, in Applicant's representative's opinion, neither Liu nor Shankman are even remotely related to Applicant's claimed invention. Neither reference alone, or in combination, teaches, mentions, or suggests Applicant's claimed shielded test execution engine. Neither mentions or suggests operating-system or hardware shielding of a software routine.

Applicant respectfully submits that all statutory requirements are met and that the present application is allowable over all the references of record. Applicants do not believe that further prosecution will produce any additional references that alone, or in combination, will come any closer to providing a reasonable basis for an obviousness-type rejection than those already provided by the Examiner. Therefore, Applicant respectfully requests that the current application be passed to issue.

Respectfully submitted,
John S. Eden
OLYMPIC PATENT WORKS PLLC

By _____
Robert W. Bergstrom
Reg. No. 39,906

Olympic Patent Works PLLC
P.O. Box 4277
Seattle, WA 98104
206.621.1933 telephone
206.621.5302 fax

APPENDIX

1. A testing platform within a computing resource environment, the testing platform comprising:

a test execution engine that receives input commands and initiates processing of the input commands;

a test routine; and

components that serve to adapt hardware and software interfaces of the computing resource environment to the test execution engine and test routine and that shield the test execution engine and test routine from dependencies on the hardware and software interfaces of the computing resource environment, including the operating system interface.

2. The testing platform of claim 1 wherein the hardware and software interfaces of the computing resource environment include interfaces to external hardware components and peripherals, external data storage and data I/O devices, communications hardware, operating system interfaces, and software interfaces to programs and routines.

3. The testing platform of claim 2 wherein the components that serve to adapt hardware and software interfaces of the computing resource environment to the test execution engine include:

user I/O components that adapt the test execution engine to user I/O interfaces;

result handling components that adapt the test execution engine to data output and presentation interfaces, including interfaces to disk files, printers, and databases; and

computing resource components that adapt the test execution engine to operating system interfaces, including memory management interfaces and timer interfaces.

4. The testing platform of claim 2 wherein the testing platform can concurrently include and employ multiple user I/O components, multiple result handling components, and multiple computing resource components.

5. The testing platform of claim 2 wherein the components that serve to adapt hardware and software interfaces of a computing resource environment to the test routine include:

user I/O components that adapt the test routine to user I/O interfaces;

result handling components that adapt the test routine to data output and presentation interfaces, including interfaces to disk files, printers, and databases;

computing resource components that adapt the test routine to operating system interfaces, including memory management interfaces and timer interfaces; and

communications components that adapt the test routine to communications interfaces.

6. The testing platform of claim 5 wherein the testing platform can concurrently include and employ multiple user I/O components, multiple result handling components, multiple computing resource components, and multiple communications components.

7. The testing platform of claim 1 further including components that adapt the test execution engine to the test routine and that adapt the test routine to the test execution engine.

8. The testing platform of claim 7 wherein the components that adapt the test execution engine to the test routine include:

a test executor component that adapts the test execution engine to a test routine linked to test platform object code in a common executable;

a test executor component that adapts the test execution engine to a separate test routine executable that runs within the computing resource environment; and

a test executor component that adapts the test execution engine to an external test routine.

9. The testing platform of claim 7 wherein the components that adapt the test routine to the test execution engine include:

test link components that adapt the test routine to user I/O components, result handling components, computing resource components, and communications components.

10. The testing platform of claim 1 wherein multiple test routines can be concurrently handled by the test platform, and may be executed:

concurrently;

sequentially;

synchronously;

asynchronously; and

according to programmed execution patterns.

11. The testing platform of claim 10 wherein a test sequencing component handles launching and execution behavior of groups of test routines.

12. The testing platform of claim 1 wherein a mode component interprets user input as testing platform commands and dispatches appropriate routine calls to execute testing platform commands.

13. The testing platform of claim 12 wherein multiple mode components are included in the testing platform, with a single mode component active at each instant in time.

14. The testing platform of claim 13 wherein deactivation of a mode component and activation of another mode component may be elicited by an input command.

15. A method for flexibly, extensibly, and portably testing components, the method comprising:

providing a component to test;

developing a test routine that tests the component, shielding the test routine from dependencies on hardware and software interfaces, including the operating system interface, by employing interfaces to adapter components within the test routine; and

running the test routine from a testing platform that includes a test execution engine shielded from dependencies on hardware and software interfaces by employing interfaces to adapter components within the test execution engine.

16. The method of claim 15 wherein employing interfaces to adapter components within the test routine further includes:

employing interfaces to user I/O components that adapt the test routine to user I/O interfaces;

employing interfaces to result handling components that adapt the test routine to data output and presentation interfaces, including interfaces to disk files, printers, and databases;

employing interfaces to computing resource components that adapt the test routine to operating system interfaces, including memory management interfaces and timer interfaces; and

employing interfaces to communications components that adapt the test routine to communications interfaces.

17. The method of claim 15 wherein employing interfaces to adapter components within the test execution engine further includes:

employing interfaces to user I/O components that adapt the test execution engine to user I/O interfaces;

employing interfaces to result handling components that adapt the test execution engine to data output and presentation interfaces, including interfaces to disk files, printers, and databases; and

employing interfaces to computing resource components that adapt the test execution engine to operating system interfaces, including memory management interfaces and timer interfaces.

18. The method of claim 15 further including:

employing interfaces to adapter components within the test execution engine that adapt the test execution engine to the test routine; and

employing interfaces to adapter components within the test routine that adapt the test routine to the test execution engine.

19. The method of claim 18 wherein employing interfaces to adapter components within the test execution engine that adapt the test execution engine to the test routine further includes:

employing interfaces to test executor components that adapt the test execution engine to a test routine linked to test platform object code in a common executable;

employing interfaces to test executor components that adapt the test execution engine to separate test routine executables; and

employing interfaces to test executor components that adapt the test execution engine to external test routines.

20. The method of claim 18 wherein employing interfaces to adapter components within the test routine that adapt the test routine to the test execution engine further includes:

employing interfaces to test link components that adapt the test routine to user I/O components, result handling components, computing resource components, and communications components.

21. The method of claim 15 further including employing within the test execution engine an interface to a test sequencing component that handles launching and execution orderings and synchronicities of groups of test routines.

22. The method of claim 15 further including employing within the test execution engine an interface to a mode component that interprets user input as testing platform commands and dispatches appropriate routine calls to execute testing platform commands.